Алгоритмы на массивах

Алексей Владыкин

СПбГУ ИТМО

12 октября 2009

Представление массива в памяти

- Одномерный массив — непрерывный участок памяти
- Двумерный массив — либо один непрерывный участок, либо массив указателей на отдельные одномерные массивы
- N-мерные массивы — те же два варианта, но на каждую пару соседних измерений

Адресация элемента массива

- a[N]:

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

- a[M, N]:

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

- a[M, N]:

$$addr(a[i,j]) = addr(a) + (i \cdot N \cdot +j) \cdot sizeof(type)$$

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

- a[M, N]:

$$addr(a[i,j]) = addr(a) + (i \cdot N \cdot +j) \cdot sizeof(type)$$

- a[L, M, N]:

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

- a[M, N]:

$$addr(a[i,j]) = addr(a) + (i \cdot N \cdot +j) \cdot sizeof(type)$$

- a[L, M, N]:

$$addr(a[i,j,k]) = addr(a) + (i \cdot M \cdot N + j \cdot N + k) \cdot sizeof(type)$$

Основные алгоритмы

- Обмен двух элементов местами
- Последовательный просмотр
- Поиск в упорядоченном массиве
- Циклический сдвиг
- Сортировка

Обмен

Встречается во многих алгоритмах.

```
t = x;
x = y;
y = t;
```

Еще варианты?

Обмен

Встречается во многих алгоритмах.

```
t = x;
x = y;
y = t;
```

Еще варианты?

```
x = x ^ y;
y = x ^ y;
x = x ^ y;
```

```
x = x + y;
y = x - y;
x = x - y;
```

Последовательный просмотр

- Просмотр всех элементов по порядку, сложность $O(N)$
- Вывод массива на экран
- Вычисление суммы элементов, среднего значения
- Поиск минимума, максимума
- Поиск заданного значения
  (и вариант реализации с барьером)

Listing 1: C

```c
for (i = 0; i < N; ++i) {
    // do something with a[i]
}
```

Listing 2: PHP

```php
foreach ($array as $item) {
    // do something with $item
}
```

Поиск в упорядоченном массиве

- Некоторые задачи можно решить быстрее
- Поиск минимума, максимума, k-й порядковой статистики — прямой доступ за $O(1)$
- Поиск заданного значения — двоичный поиск за $O(\log_2 N)$

Поиск в упорядоченном массиве

- Некоторые задачи можно решить быстрее
- Поиск минимума, максимума, k-й порядковой статистики — прямой доступ за $O(1)$
- Поиск заданного значения — двоичный поиск за $O(\log_2 N)$

```
int l = 0, r = N;
while (l < r) {
    int m = (l + r) / 2;
    if (a[m] < x) {
        l = m + 1;
    } else {
        r = m;
    }
}
if (r < N && a[r] == x) {
    printf("found %d at a[%d]\n", x, r);
}
```

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Первый алгоритм:

- k раз сдвинуть массив на один элемент влево — $O(Nk)$.

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Первый алгоритм:

- k раз сдвинуть массив на один элемент влево — $O(Nk)$.

Второй алгоритм:

- сдвинуть $\gcd(N, k)$ цепочек на k влево — $O(N)$.

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Первый алгоритм:

- k раз сдвинуть массив на один элемент влево — $O(Nk)$.

Второй алгоритм:

- сдвинуть $\gcd(N, k)$ цепочек на k влево — $O(N)$.

Третий алгоритм (Керниган и Плоджер):

- перевернуть $a[0..N-1]$, затем $a[0..N-k-1]$ и $a[N-k..N-1]$ — трудоемкость $O(N)$.

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Первый алгоритм:

- k раз сдвинуть массив на один элемент влево — $O(Nk)$.

Второй алгоритм:

- сдвинуть $\gcd(N, k)$ цепочек на k влево — $O(N)$.

Третий алгоритм (Керниган и Плоджер):

- перевернуть $a[0..N-1]$, затем $a[0..N-k-1]$ и $a[N-k..N-1]$ — трудоемкость $O(N)$.

Четвертый алгоритм (кольцевой буфер):

- назначить $a[k]$ началом массива — трудоемкость $O(1)$.

Упражнения

Обойти прямоугольную таблицу $a[M, N]$:

- по строкам;
- по столбцам;
- горизонтальной змейкой;
- вертикальной змейкой;
- диагональной змейкой;
- по спирали.